

Kevin Dallas: Powering Smart, Connected, Service Oriented Devices

Remarks by Kevin Dallas, General Manager, Windows Embedded

Embedded Systems Conference East 2007

Hynes Convention Center

Boston, Massachusetts

September 20, 2007

Kevin Dallas: Good afternoon, everybody. I appreciate the opportunity to come and talk to you about the direction that we're taking at Microsoft around the Windows Embedded Business, and over the next 45 minutes, I'm hoping to describe for you our focus on connected experiences, and how smart, connected, and service-oriented devices are really going to power those experiences. With three billion embedded devices forecast to be shipped this year, it's pretty clear that it's a tremendous business opportunity for many of you in the room, and it's also a big opportunity for us at Microsoft. As you can see, there's just a boom in smart devices that are IP connected, and these are devices that are connected to a wireless or a fixed line network. You can also see that there's just some phenomenal growth rates that we're seeing across these different devices.

In the PC space, we're seeing about 4 percent growth; on laptops about 19 percent growth. This kind represents the core of Microsoft's business today. But then we see a tremendous opportunity around a new class of connected devices. These are connected enterprise devices like a thin client that may connect to a network, to a management service, or a connected consumer device, like a PND device that connects back to a navigation or map set of information. And you can see the growth rates: 23 percent on enterprise devices and 50 percent on consumer devices. What's interesting is, out of the three billion devices that will ship, about two-thirds of them are going to be connected, and that's why we as a company are focused on building up the platforms, the tools and services so we can help our partners and customers address this growth opportunity.

I'll give you a couple of examples on some experiences where we could improve as an industry. Let me start off with communications. For example, today if you receive a call in your home, that call will usually be on a land line. If you go out into the Wide Area Network, you'd have to hang up, call the person back on your mobile phone, and if you get into your corporate headquarters, you'll normally have to hang up again, pick up a separate phone, your IPBX phone, and then return that call. So you're literally on three separate phones. When I talk about a connected experience, in the future, what you'll see is the ability to receive that call in your home on your mobile device, it knows that it's sitting on a Local Area Network and will use VoIP for

that communication. Then as you move toward traveling to the office, you move toward your car, you switch seamlessly to a Wide Area Network, and as you get into the car, the hands-free capabilities then instantly switch in. As you continue that call and you get to the office - this is a very long phone call, by the way - you expect that call to then seamlessly switch over to your IPBX, so your mobile phone should recognize that you're now on your corporate network, and will use that corporate network for that call. As a user, this is a very seamless, connected experience that went straight from the home environment to the mobile environment to the corporate environment.

Another example would be around navigation. Today, if you're on your PC and you want directions to a restaurant, you'll look up those directions, you'll then have your PND device, your portable navigation device, download it to the device, or even worse, if you want that information in your car, you have to go to your car and manually put that information in. In this connected experiences world, what should happen is you should go in, type the information that you need on your computer, have that information seamlessly go into your personal cloud, and then be able to access that, irrespective to the device that you're using. So your portable navigation device should know that it should synchronize up to that cloud to get that information, or the navigation system in your car, should be able to go to that cloud to pull it down. So as you get into your car, that information is there and available and ready to guide you to that restaurant.

These are the types of connected experiences that we as a company are very focused on, and you can see that in a lot of the products that we're coming out with, especially in the embedded space, where my business unit is really focused. So you'll see a number of connected entertainment experiences, connected information sharing experiences, all the way through to productivity and learning. I've talked about the high level experience and what that means on the end user level.

The next level down is what Microsoft and its partners are doing to enable this new world. This is where the smart, connected, service-oriented devices come in, and I wanted to spend a bit of time just talking about what we mean by service-oriented devices. These are devices that are going to be service aware, they're going to be able to reach out and access and discover Web Services, but in addition to that, a lot of these embedded devices have some very important information that's sitting down on the device in such a way that if you could expose that information, it would be very valuable for an OEM customer, an enterprise customer, or a service provider, in terms of delivering additional value to that end customer.

Now, I'll give you some examples of this. On the management side, if I were able to connect to a consumer device, be it a set-top box or digital video recorder, the manufacturer would be able to proactively monitor that device, determine when inputs are needed for that device, and transparently send those updates down to that device, as opposed to having the consumer see a problem on the device and then place that call. Another opportunity is to actually remotely monitor the device in terms when do we need to replenish this device. I'll give you a couple of examples here; in the enterprise space, if you have a multi-function printer, if you could see when that toner needed to be replaced, and before the customer asks for it, be able to deliver that new toner for that multi-function printer, that's a service that an OEM could actually provide to an enterprise customer.

Another area is usage. We have the ability in these devices with the operating system, to be able to monitor how the device is used. So you can imagine a scenario where the end customer gives the manufacturer the ability or the permission to monitor how that device is used. Two opportunities then are put forth to the OEM or service provider: First, it can monitor how the device is being used, and as a result present new services to the customer, which will drive an even improved service for the customer as well as a revenue opportunity. The second is more of a subtle piece of feedback, which is, as we're developing our products, we're all challenged in terms of really understanding what kind of functionality, what kind of scenarios our customers want. So by having this kind of capability, you're able to pull back information in terms of how the customer is using the product, use that information to define an improved next generation product as well. Location, I think that's an obvious one, where with these devices you can be able to determine its location, which is also important information for an enterprise. And then advertising is another very interesting and high growth area, where a lot of devices are shipping today, they may not ship with an advertising capability, but in the future, if it's a service-oriented device, you should be able to reach out to that device, determine whether it's capable of displaying advertisements, and if so, be able to publish advertisements on that device. Again, it's something the subscriber would subscribe to, and opt in on, but it would give the OEM or the service provider an opportunity for an improved service and a new business model.

So these are the device services that I'm talking about. Not necessarily the ones that are sitting in the cloud as Web Services, but I'm saying there's a bunch of rich information that sits down on the device that we can make sure is accessible to the OEM, enterprise customers, service providers that can then use those to deliver more enhanced services for the customer as well as drive revenue. So again, service-oriented devices is a very big focus for us. So as we're thinking through this future world of tomorrow, there are some

very important trends that we are understanding and frankly trying to design our solution around. The first one is the Software Plus Service. Ray Ozzie, our chief software architect, talks about this all of the time at Microsoft, to the point where the products that we're building really do have this core technology trend in mind, which is that we can no longer build software that just sits on a device. We need to build software plus a service. We build a software that will run on the device but then in most cases, we're starting to see that device have a very important service element to it as well. Now you've seen that with the Zune device, where you have an operating system that sits on the device, but there's a very important service component in terms of the music service.

We have digital video recorders, great devices, but they have a very important video on demand type of service that would connect to that. So what we need to do in the embedded space as we build out our products is make sure we build up that core software platform, but also think about the service components that are needed to complete that connected experience. For example, with many core, if you look at most of the embedded devices today, they usually have a single processor on them and this single processor is really programmed with low-level code, sometimes high-level code, and what we're trying to do to get the performance up on these devices, is we've been clocking these CPUs at a faster and faster rate. And we're fast approaching 3 gigahertz now on one of the high-end CPUs. But what the CPU manufacturers are finding is that we're rapidly hitting the bar where it's hard to get increased performance without driving up power consumption and of course heat dissipation; it's a really big issue. So a lot of the advanced CPU manufacturers are now moving to world of many core, and this is many core in a very heterogeneous core environment. So think of this as instead of clocking the frequency higher and higher, we're moving to this world where you'll have multiple heterogeneous cores on a single chip. And this could be an applications processor, graphics processor, radio processor, all on a single chip, where you're able to drive performance, and then keep the heat and power consumption down. This does present challenges when you're writing these applications, it will need to run across these cores and I will get into that at a later point.

Next is a loosely coupled service. Today, a lot of the solutions that are out there are tightly coupled - you have media devices that are tightly coupled to a music service. You may have a set-top box that's tightly coupled to a video-on-demand service, or you may have a Voice over IP phone that is tightly coupled to a Voice over IP service. In the future, these devices will need to be able to connect- discover and connect to other services. So you'll buy the device for a primary function, but as you start to evolve this connected experience, you want this device to be able to actively go and discover these

other services and be able to run them on the device. So there's another aspect here that I want to emphasize is that not only do we need to design for these tightly coupled, but also make sure that these embedded devices can connect to these loosely coupled services and run them if they have the capability.

And then, of course, there's the service-oriented development. So imagine this environment where you now have these devices that are able to participate in these connected experiences. And then behind these devices, you have a whole set of services that this device is now connecting to and leveraging. These services may be on the same device in another core, it could be on the circuit board, it could be on the premises where the device sits, or it could be connecting out in to the cloud, in to a web service that's, frankly, on the other side of the world. So the question is how do you build a rich application that's going to be able to abstract you as a developer away from all these different services? So you actually create this rich application that runs on your service-oriented device.

These are really the four trends that we see and we've put a lot of energy in these areas, and we're going to be showing you a demonstration in a minute especially on the last point. So for us, the focus is on connected enterprise experiences, so as we think about what our software needs to do to power the next generation of thin clients or the next generation ATM machine or the next generation point-of-service terminal. We're not just thinking about the operating system that needs to sit in that device, we're thinking about what the experience needs to look like in terms of manageability on a thin client solution, and then what do we, as a company, need to provide in terms of the platform to enable our customers to deliver on that proposition? We're also looking at connected consumer experiences. As you know, our platforms are used in portable media players, set top boxes, PND devices, so we're focused again on not only the operating system that sits in that device, but also what kind of services to we need do connect to fulfill that end to end experience?

There's also some work that we've been doing for a number of years with our Microsoft Research Group around a service-oriented development environment. We realize that it's not enough to build these components into our platform, it really comes down to giving the developers the power through this development environment to actually create these connected experiences. So we've actually made a big investment in this area and I'll be demonstrating that in a minute. Then, of course, once you have all of these devices, it's really critical that you're able to manage them. It's not enough to just have those devices out there connected to services, you need to be able to monitor that and be on top of it; if there are issues, be proactive in terms of correcting those issues, and then you need the service-oriented management service.

With that, I wanted to spend a few minutes demonstrating some work that we have done around the service oriented development environment, and I'm going to invite Mike Hall, our software architect that many of you may know, up to the stage to give us a demonstration. Mike.

Mike Hall: Thanks, Kevin. As Kevin mentioned, we've been spending some time working with tools with the Microsoft Research Group to better enable the development of service-based devices. There are some very clear trends that exist for embedded developers both at the hardware level, but also at the software level as well. When I think back to some of the work that I was doing back in the late '70s on 8-bit systems, everything that I was doing was written in assembler and it took forever to write, and the code took forever to test and get right as well. So the trend is obviously moving away from extremely low-level development tools, assemblers, and so on, to C and C++. The majority of the embedded system software today is written in C and C++. But also moving to higher-level languages such as Java and C#. Why does that trend exist? The trend exists for a number of different reasons.

First of all, an application developer that's working at a much higher level is much further abstracted from the underlying hardware and the underlying operating system that is running underneath it. The developer can be far more productive, in fact, getting much closer to rapid application development, producing more robust applications, more secure applications, faster than using a lower level language such as C and C++. In the C, C++ world, it's very easy to write application code, it's very easy to write application code that exhibits behaviors such as buffer overruns, that leaks objects, that leaks handles, and of course you need to test for that. As you move much higher up the application development stack, you move away from those issues and into an area that you can rapidly develop applications.

If we have a look at the hardware trends that Kevin was talking about just a few seconds ago, we can clearly see the trend of moving from single core devices, perhaps with single threaded applications or more complex applications, multi-threaded applications where synchronization between threads also then starts to become an issue. But then moving to multi-core and many core devices where synchronization across cores then becomes an issue. Perhaps even moving out further to distributed cores or network devices, again, writing massively distributed, networked, multi-threaded applications is a hard challenge for developers today. Many developers, when they think about building embedded operation systems, think about building these operating systems in terms of technology stacks. You know, I need a web server on this device, I need a [unintelligible] stack on this device, maybe I need IP v4, maybe I need IP v6. What we're suggesting is that you start to think at a much higher level, much closer to the move from C, C++ to these

high-level languages of Java and C#. That's on the application development side, but what about building single core, service-based embedded systems, or many core or distributed core embedded systems? So what I'd like to do now is just give you a demonstration of a tool that has been developed inside of Microsoft Research which is known as VPL or visual programming language.

[Demo]

On the left-hand side of the screen, you'll see that we have a number of services listed. There is no component or definition with these services at this level of the underlying technologies or operation system components needed for these services to run. So as a developer I can build a single core based design, a many core based design, or a distributed core based design by linking services together and then making the determination at build time about where those physical services are going to run on my single device or my distributed devices.

Kevin Dallas: Mike, these services that you have on the left-hand side, I assume that some of these can reside on my local device, on my single core or multi-core, or it could be on another embedded device on my premise or it could be a Web Service that exists up in the cloud.

Mike Hall: Exactly right. So let's go ahead and build a sample application. What I'm going to do here is just grab one of these services out of the catalogue and this service is known as a color segment service. This is a service that is looking for frames or images coming from a camera and is looking for very specific colors inside of the images that are coming back from that camera. I can configure this color segment using the properties on the right-hand side to import an XML file that will define the camera and the colors that I am looking for. What I can then start to do is start to add additional services into the mix. In this case, I'm going to add an if statement, the output from the color segment service could be that a number of colors have been found and a number of color segments have been found. So I can start to link some of these components together, linking the color segment to the if statement. So I can say that if the number of color areas has been updated, then fire this if statement service. Within the if statement service, what I want to look for here is the areas item, and the count of areas being greater than zero. If the number of areas found, in this case of a red color, within the images coming from a camera is greater than zero, then I want to link this if statement into a data service. And in this case, I'm simply going to change its type to a string and say that I've found red. And the output of this could well be linking into a text to speech service, it could well be that my initial embedded system that I am building for here has the capabilities of text to speech, but if it doesn't, maybe the text to speech engine could reside

on a completely different node the network, and in fact running a completely operating system image to the core service that is looking at the color segment in the camera.

Kevin Dallas: Mike, if I look at each of these service nodes, how much do I need to know about each one of them? Do I need to understand what the underlying operating system is, what kind of capabilities it has, and how much knowledge do I need to have?

Mike Hall: Okay, that's a great question. But from the capabilities that you've seen so far, you don't need to know and understand the underlying operating system image that is needed for the service, or in fact the underlying capabilities of that service. As you start to link the inputs and outputs of services together, you are dynamically prompted for the inputs and the outputs that you want to match. So in some cases you may try and take the output from a color segment service, or in fact from a webcam directly, which, of course, would be an image type, and link that into a string. Of course, that's not going to work, so the tools will be smart about determining the match up between the output of one service and the input of another service.

Kevin Dallas: So this is very much a cross-platform environment, so I could have different types of operating systems, different types of service systems that can all be abstracted away using this tool.

Mike Hall: Exactly right. Okay, so what I'm going to do here is find a text to speech service. I'm going to link the data found red directly into this text to speech service and I'm going to say, Take the data value that's passed in, use the text to speech service to say this text, and what I want to do here is say the value of the text that gets passed in. So in this case, that's it, the core application that I wanted to build is now done. I've not defined the underlying operating system or operating systems, but I have defined the data flow between the various services that I've mapped out on the screen. So this is a very simple example of an application. Let me just minimize this and move over to a slightly more complex version of the same application which makes use of two separate cameras, and in fact, it just so happens that I have two cameras up here on the stage. These could be completely different nodes running on different cores within the same machine or across many cores distributed across the network.

Kevin Dallas: Okay, Mike, so let's pause for a while. That took you a few minutes to create I assume.

Mike Hall: This was less than 10 minutes to create this application.

Kevin Dallas: Ten minutes. But then I'm sure many people in the audience would like to know, and I would like to know, what does it actually take to implement this? You're going to have all these distributed systems but what kind of work do you need to do to make sure that these services actually communicate and deliver this application?

Mike Hall: Okay, that's a great question. So we have a number of services that exist in the catalogue here ranging from text to speech and camera support and color segment, and some of the switching statements if and else and merge and these kinds of things. These are example services that we've created to show you some of the capabilities here. Within Visual Studio 2005 you can also create your own custom services. The interesting thing here is that the communication between services is just based on standard industry standards such as HTTP and [unintelligible] protocols. So actually the interface between each of these services is very well known and it's very easy to implement.

Kevin Dallas: Okay, so this all based on industry standards, there's nothing proprietary here, this can be used across any service.

Mike Hall: Exactly. Okay, so let's go ahead and run this application. One thing that you'll note is that the application does not expose any kind of user interface. So we won't actually see any kind of images or dialogues or any kind of thing coming up on the screen apart from the information from the underlying run time showing which services are getting started. That doesn't necessarily mean that the services can't display user interface. A service could very easily display any kind of user interface that it wanted. This could be a window, this could be a message box, this could be a full direct 3D kind of surface with rotating images or whatever else. So I've launched the application, but of course, you want to know that the application is running right? Okay, so I've mentioned that the interface into services is HTTP and [unintelligible], that means that using a standard Web browser, I can take a look at the status of any of the services that are running at this time.

I can just click this link here, this will launch Internet Explorer and give me a view onto all of the services that are currently running as part of this application. Once I get this application map, and you can see the same diagram shown at the bottom of the screen here, I can then link in to the running services directory and take a look at some of the services that are currently running, in this case the color segmentation service. So this shows me on one side the camera, and you can see my hand coming in here, and on the right-hand side, the segmented service, which in this case is looking for the color red. At the moment I'm not holding anything red up in front of that

camera so it's not reporting anything. But it just so happens that I have a red book here. As I start to hold that up, you can see the color segmentation service is picking up the red color and highlighting that within the color segmentation area. I'm not real sure whether you heard it or not, but the text to speech service on the laptop also fired in and said that is saw something on the left. Well, that's obviously your left, my right. So that's a quick view of the application running and a way for you to view the application running using a simple Web browser, giving you the ability to look at the service instance running on a particular node, which, of course, could be on any machine that's on the running network and to change, modify, and view the capabilities of that running service. Okay.

Kevin Dallas: Thanks, Mike. Okay, so what you saw there was a tool that we've actually been working on for a number of years at Microsoft in the Research organization, and we're now bringing that into the business group of Windows Embedded Team and we're doing work on commercializing this as a product. What this will allow us to do is, as Mike outlined, you'll be able to build applications that may be simple text or message boxes all the way through to complex applications that would sit on top of these distributed services. Again, big investment for us, and something that you will see in the near future.

So I've talked about tomorrow and where we're going with connected experiences at the user level, the smart, connected service-oriented devices. I wanted to switch gears a little bit and talk about what we're looking to deliver here and now from a product line perspective. We have a complete line of Windows Embedded products, all the way from the low end with the .NET Micro Framework, all the way to the high end where we have our XPE operating system, as well as a license restricted Vista operating system that I will talk about in a minute. In addition to that, we have the Visual Studio Tools and a huge developer base. You can see that Visual Studio is arguably the most popular integrated development tools environment in the industry, and we have more than six million developers that are familiar with that tool set and can actually develop applications and services for the embedded environment. We also have an ecosystem of partners and I would argue that this is probably the most important component here in our product stack where we have a three-tier partner program where we have our baseline partners, Silver, and Gold that are there to support our customers in getting to market with their embedded devices. We have about 1,500 partners in this partner ecosystem.

Then last but by no means least is this new element which is services. We've made investments inside of Microsoft around Live services, I won't go in to too much detail there around the investments that we're making, I think you'll

hear that in separate presentations, but I also wanted to highlight here that it's important that our embedded devices can connect in to third party services as well through standard interfaces and standard protocols, and that is indeed the investment that we're making. So this is really the infrastructure we're putting in place really to power these next generation devices.

Let's walk you through the product line. I think I'll start off with the Windows Embedded for point-of-service product first. This is based on XPE Service Pack 2, and what we've done is we've taken specific steps to optimize this solution for a point-of-service terminal. So we've made optimizations in terms of the way the set up is presented, the macros that you can use to create a reduced footprint solution for a point-of-service terminal. And we have it adopted by some key partners, like Hollywood Videos, Virgin Megastores, so it's a broadly adopted platform and this platform has really been in the market for a couple of years for us, so it's a relatively new platform but we're seeing tremendous adoption behind it.

We have the XP Embedded product line, and I'll spend a bit of time on this because I know that we may have not been as clear as we should have been around where we're going with this product, so I want to spend a bit of time on this product in particular. For those that are not familiar, this is based on our XP operating system and with it comes a full infrastructure that you have in the PC space in terms of driver compatibility, hardware compatibility, so it allows our customers to take this platform and get to market very quickly using that PC infrastructure. The tool and framework that you use in a PC today are the same tools and framework that can be used on the XP Embedded platform as well. The other component here is really the optimizations that we've made for an embedded, so we went in and we've componentized that operating system so that we have approximately 3,000 operating system components and about 6,000 driver components that you as a developer can pick from to create your optimized solution. This allows you to really optimize in terms of footprint, it allows you to reduce the attack area from a security perspective and also you have the powerful tools that will then go with it.

In terms of footprint of these embedded devices, it can really range depending on the solution that you're trying to build, but you can get down as low as 128 meg in terms of size and go up from up there, and what we try to do is to show a typical range for the lower end XPE solutions. I've talked about being able to optimize the attack area from a security point of view, which is one of the reasons why a number of our customers select this solution versus the desktop solution XPE. Then we have some embedded enhancement features that many of our customers had asked for that we have now delivered, there's

the USB boot feature that we've added, and there's also the enhanced write-based filter technology that we've added to this product.

From a road map perspective, I will highlight a few things here. We came out in 2006 with our feature pack and we highlighted a couple of features that I've talked about: the [unintelligible] filter, USB boot. A lot of people have asked questions around what are we doing next? What's going to happen in 2008 in terms of the solutions and features we support, and then frankly what's going to happen in the Vista time frame is another question. In 2008 we are coming out with another version of our Embedded platform which would be based on XPE Service Pack 3 and then we're also looking to pull back the enhanced Vista features that are applicable for the embedded space. So we spent some time with our customers asking what kinds of features they need for their embedded systems and the feedback has been that we need the updated browser, we need the updated Media Player, Winamp 11, we need the RDP, client RDP 6, and we also .NET 3.0. So those sets of functionality is what we're bringing in to the XP Embedded platform and we'll be delivering that in 2008.

The next question, what's happening with Vista? So today in your product line, we have a Vista solution, which is, frankly, the full desktop solution that can be licensed for an embedded environment. The difference here is it's not a componentized solution, it's the full Vista solution, which is satisfactory for certain applications like ATM machines, but there are some applications that need more of a componentized implementation of Vista, so we spend time talking to our customers and partners, and the feedback that we've got is that they would like to see that componentized solution in the 2009/2010. So we are committed to delivering a componentized version of Vista in this time frame, so we'll be moving from an embedded product based on XPE to an embedded product based on a componentized Vista, which is a very important clarification that I want to make sure that everyone understands. So again, we'll stay with XPE, we'll move forward in 2008, we'll bring into that product the advanced Vista technologies and then in 2009, 2010 you'll see the componentized Vista product.

The next product we have is CE, and CE is a small real-time operating system. You often see this operating system in a lot of the mobile devices like portable navigation systems and also your portable media players, where you have a small footprint, you also have reduced power requirements, and in some cases you have a real time component as well. And that's when the CE operating system tends to be used. Another powerful feature here is that this operating system runs on a number of different operating systems, ranging from X86 and on, all the way to MOS and SH. It's a highly customizable product, it has a componentized architecture very similar to XPE, so you as a developer can

just pick the components that you need for your design. Another point that I'd like to emphasize here is the shared source program that we have, so we've opened up CE as a shared source to developers, which gives you the ability to take the product, make your optimizations, add your own value, ship that and commercialize it without the need to then provide that back to Microsoft, which is a very, very, very important differentiator for you as our customers.

Just to give you a sense of where we're going with the product, we view it internally as three components, there's the kernel, there's the components, and there are the tools. You'll notice here that every two years, every 24 months, that's when we do a major release, that's when we really touch the kernel, the components, and the tools. Then in between, what we do is we use that as an opportunity to be innovative and deliver the differentiation that our customers need, and it may be innovation that's need at the component level, or it may be innovation that's needed at the tools level. So you can see here we've got CE 5.0 and 6.0, which were the major releases where we delivered the kernel, the components and the tools, and then on 5.0, we used that as an opportunity to deliver the network media device solution, which is more of a component differentiation for our customers. Then on 6.0 with the platform builder service pack, you'll notice here we're focused more on the tools environment and what we need to do add more value. So in November, what we're looking to do is to deliver another update which will have some more differentiated components for our customers.

What I would like to announce today is that we are going to be launching Windows Embedded CE 6.0 Release 2 in November, November 15, in Japan at the largest event, embedded event, in the world. This solution has a number of very interesting components that I'd like to share with you; I'll just kind of give you a sneak peek of what we're doing. Just some very exciting technologies here, and I'll also try and spend some time, especially in the demo, connecting what we're doing here, with where we're going around service-oriented devices. Three areas that we're going to share with you today, there are many, many more components that we're adding so our customers can really differentiate. But there's updates in the browser that we're delivering around Ajax support in the browser, around heavy data entry support in the browser, so we're adding those types of features. There's also a video- Voice over IP, is another big area as you know in the PC environments as well as standalone phones, so we're adding that kind of functionality to enable our customers to deliver these Voice over IP scenarios where we're not only building in functionality that allows you to do Voice over IP, but we're supporting advance features like 3-way audio as well as video conferencing in this stack as well. So it's a very advanced solution that we're delivering on the Voice over IP front.

Then we have our Web Services on devices, and this is really the first step toward our service-oriented devices vision, and you'll see that this is really going to provide the first step in the foundation for us as we deliver on this vision of service-oriented devices. So with that, I'm going to spend a bit of time demonstrating this to you, so we're going to ask Mike to come back on stage and demonstrate to you the Web Services on devices component of Windows CE 6.0 R2.

[Demo]

Mike Hall: So when Kevin's talking about Web Services on devices, this is Microsoft's implementation of the industry standard DPDS or devices profile for Web Services. And what I'd like to talk about and show in just a second is how that implementation of WSD or Web Services on devices can be used to communicate between devices and between devices and PCs.

Kevin Dallas: Quick question. So Web Services on devices, is that a Microsoft term or is that an industry term, can you put that into perspective?

Mike Hall: So the industry term devices profile for Web Services or DPWS. WDS or Web Services on devices is Microsoft's implementation of that standard.

Kevin Dallas: Okay, so this is an industry standard that we're implementing?

Mike Hall: Yes, correct. Okay, so I'm sure that many of you in this room have been in the situation where you've taken a laptop into a meeting room somewhere and you're wanting to project the contents of your laptop screen out through a projector onto a display surface. Has anyone ever been in that situation? Yes, one or two, okay good. So what you've probably found is that you've gone into the meeting room, you've got your laptop, you get this snake from the back of the projector, you plug it into your laptop, you press various function keys, you change screen resolution and refresh right, you snap your heels together and spin round on your chair and eventually the image changes to match the capabilities of the projector.

Kevin Dallas: This is Mike watching me in a conference room, by the way.

Mike Hall: Yesterday in fact. So in that situation it's kind of hard work and you're trying to match the video output capabilities of the laptop to the video input capabilities of the projector. With Windows Vista, we shipped a network projection Wizard, which gives you the ability from a Vista-based laptop to do a search of the local network for a network-based video projector. Once

you've found that projector, you simply connect that projector, there will be some interfacing between the Windows Vista part and the network projector part, and without any cabling, without any function keys, without any changing of screen resolution, you can directly output your laptop display surface on to that video projector. The underlying technology that makes that happen is Web Services on devices, so that's a really good example of technology that ships today that gives you the ability to link a device, in this case a projector, to a laptop which is running Windows Vista.

Kevin Dallas: So what you're about to demonstrate is the Web Services on devices which is communication device to device or node to node. But does this also include connecting out to a web service, or is that a separate phase?

Mike Hall: Okay, so in this situation, you can have different nodes taking part in the conversation, you can have multi-nodes talking WSD to each other. So that means that- take a home environment, something like home automation, you can have a home automation controller that talks out to your heating system, to your lighting system, to your blinds and so on. So you have multiple nodes coming on and off of the network reporting to your central remote controller that they're available and the services that they provide. Each one of these nodes on the network can of course independently call out to XML Web Services, so you can easily bridge that gap between WSD and XML Web Services. Okay, so why don't we take a look at how we implement an XML- a Web Services on devices, or WSD-based application inside of CE 6 R2 operating system. So what I have here is Visual Studio 2005 and the CE 6 development tools are integrated into the Visual Studio 2005 product. What I'm now going to do is just configure a baseline CE 6 operating image and include the Web Services on devices component as well as an application that uses WSD to talk between a CE reference board, and in this case, an application that's running up on my Vista laptop.

Kevin Dallas: So does this mean then that we're going to have an integrated environment so we're using Visual Studio for this as well and then also gives any CE device that's using R2 the ability to communicate in a scenario with a Vista PC as part of that service, because the Vista PC has it as a native stack? Mike Hall: Correct. So Windows Vista has WSD as a native stack, the software development tools are available for you to write WSD-based applications on Vista, and of course, Visual Studio 2005 is the default client for building those kinds of applications. With the CE 6 development tools plugged into Visual Studio 2005 this of course now gives you one place to generate your desktop application on this side of the equation, your operating system image for the CE 6 reference board that I have up on the stage here, as well as the application that runs on top of that reference board as well. Okay, so I'm just going to come in here and select File, New, Project, and this will

prompt me for the type of project that you want build. So this could easily be a Windows mobile application. This could be a Windows desktop application. This could be a web server based application, or this could be CE 6 operating system image which I currently have selected here. So I'm going to click okay, this will start something called the OS design Wizard, which will prompt me- ask me a number of questions about the operating system design that I want to build.

Mike Hall: So I'm going to select Next, I choose the type of hardware that I want to deploy to. In this case I'm using an [unintelligible] Vortex Board for this demonstration, but I can easily multi select various board support packages, or BSPs, as part of this Wizard. You can think of Windows CE as being two layer: the hardware independent layer, technologies such as compact framework, Web server, Web browser, Media Player and so on, and the hardware dependent portion which is the BSPs and drivers. So using these tools it makes it very easy for me to swap in and out the underlying hardware configuration without changing the higher level operating system configuration that's running on that hardware. So I'm going to step forward through the Wizard, I am going to choose an industrial device for the configuration of the operating system. I could easily choose a small footprint device, which will give me 300 kb operating system image, or a custom device, which will ask me 19 pages worth of questions about the operating system that I want to build. In this case I'll choose industrial device, and industrial controller is fine, I don't need the manage code runtime environment, the .NET Compact Framework in this instance.

Kevin Dallas: So you don't need the .NET Compact Framework? So you don't need the mänge code environment at all?

Mike Hall: Right, that's correct. So I can just write my WSD application in C, C ++ calling into the WSD APIs or WSDAPI.

Kevin Dallas: And what is the true cost of porting that stack on to my device in terms of memory footprint, any idea?

Mike Hall: Okay, well, I can give you an idea of the overall stack as it starts to build. So the CE6 kernel is about 300 kb in size. If I was to add support on top of that for networking in a web server, I get to approximately 800 kb in size. If I want to add the WSD component specifically, that's just another couple of hundred K on top of that. So you get some feel for the stacking or componentization of the operating system in terms of size. So I'm going to select Next, networking is fine, Next again; hit Finish, and that will complete my operating system design. Now, as part of configuring this operating system image, I get the security warning being displayed up in the screen.

When I build an operating system image, if there are any exposed interfaces into that device such as OBEX, Object Exchange Over Bluetooth, or the Web server or FTP server, I would get one of these security warnings telling you that you're exposing interfaces to the outside world. So in this case, there is a device host API. I can acknowledge this warning, there are links to documentation to better secure my operating system image and so of course I can read that or I can make the decision to remove this component from my operating system design.

Kevin Dallas: This is good; I get this warning, this warning that says there is potential security issue, is it also prescriptive in terms of describing for me how I get around that issue?

Mike Hall: Yes, indeed. In fact, the dialogue, we'll see it again in a second actually, the dialogue also links out to the documentation that exists around that technology and component and how to secure the operating system when you're using those technologies. Okay, so there are multiple views on to the operating system. I'm going to switch over to a catalogue items view and this contains about 700 individual operating system features, each of which are linked through a dependency chain. So for example, if I start just with kernel and layer on the HTTP Web server, the Web server, of course, needs networking, so the networking stack will automatically be included into the operating system design. What I'm going to do here is do a search for WSD API, or WSDAPI, the Web Services on devices API, which is the name of the component in the tree. And you'll see that here is the Web Services on devices component which has an exclamation point next to it. That means that there is a potential security risk by adding this component to your operating system design because it's exposing interfaces outside of the design.

So I select this component, and again, I get the security warning. Warning adding this component will expose interfaces out of your device, here is a link to all of the documentation that you need to secure your operating system image. So I'll acknowledge this, I'll move down my catalogue, and I created a component for my WSD application, which is called Bubbles Client, and you'll see why it's called Bubbles Client in just a second. So that is the configuration of my operating system design done, I can now build the operating system and deploy that operating system to my hardware. The build process unfortunately takes about 15 minutes, so we're not going to do that right now, but instead, just like good cookery show, I'll move over to a pre-built operating system image that's already running. So I'm just going to minimize Visual Studio here and I'm going to bring up a command prompt, and I'm going to run the WSD Vista side application. Up in the top right-hand corner of the screen, actually you can't see, is the unique URI, or the unique

identifier for this WSD service, and the identifier here in this case is foo. What I'm going to press a key on the keyboard which will, hopefully you can see this- a bubble on to the Vista side applications. So you can see that randomly moving around the screen.

All right, so what I'm now going to do is move over to the CE operating system image. It has the number three up on the screen because we had two rehearsals morning, so this is the third time that I've run through this demo. What I'm now going to do is just run a command prompt and from the command prompt, I'm going to type a command line, which is Bubble Client. I'm going to give it the same URI, which is foo, and give it my name which is Mike, and hit return. So this will launch the CE-based WSD application. You can see there's a dialog up on the screen that gives me a keyboard up, down, left, right. You'll see why that's important in just a second and if I switch back to my Vista machine, hopefully you can now see that there are two bubbles up on the screen, one for the computer and one for my CE-based device. So we now have event linking between the Vista side WSD application and the CE side application. So what I can do now is use the keyboard on my CE side device to move my bubble around on the screen to hopefully try and hit the Vista-based bubble. And so there's eventing that's happening from the CE side to obviously tell the Vista side that I'm moving the bubble. There's also [unintelligible] coming back from the Vista side to say that my bubble has hit the Vista side. Obviously just a couple of bubbles isn't particularly interesting, so we can just add a few more in there to kind of liven up the game slightly. So from this point of view, it's very easy for me to write using WSD a Vista side application, a CE side application and link these two applications together with eventing in both directions, and we'll be providing more information about these technologies in a session immediately after this keynote in room 107.

Kevin Dallas: Okay, so if I'm the Vista machine, how do I know that the CE machine is secure because it's trying to access the service, I need know how secure that is, how do I know?

Mike Hall: So from that point of view you have the ability as the author of that WSD application to either accept or deny connections into that Web Service, into the Web Service's device's API.

Kevin Dallas: And then through that URI any other device would be able to access that service as well, so it's openly available to other devices?

Mike Hall: Yes, so the interesting thing about URI- you'll note that this is not necessarily a machine name, this is not an IP address, this is just a unique identifier for this specific Web Service. So that means that any other device

that wants to communicate with that Web Service simply needs to know the unique URI in order to connect to and interface with that WSD application.

Kevin Dallas: Okay, thank you, Mike. Okay, so that's CE. R2, we are going to give a sneak preview of that and I'd encourage you to attend so we can actually show you more details of the great work that we're doing there. The other product that we have is the .NET Micro Framework. I think some of you might be familiar with this; this is a run time environment that takes about 400 k that runs on non-MMU-based devices. All right, so the low-end solution we had before this was CE, which needs an MMU, and this is a new addition to our family. This is a solution that is a run time environment, it extends the .NET at run time down to control-level embedded devices. It's been actually adopted by a number of our customers in the industrial automation space and the home automation space as well, and there has been considerable interest around it. It also has some of the embedded optimized features like power management, there's the managed code drive, it's actually just completely managed environment is what we have here as well as prioritized assistance.

So I've talked about the operating systems and run times that run from low end controllers all the way up to the high end. I want to spend a bit of time talking about our partner ecosystem, because again, we really believe that the success in the growth that we've seen in Windows Embedded is really due in large part to the partner ecosystem that we have; 1,500 partners worldwide, we have three levels of partners all the way through to Gold, and then we're also working with industry experts. So I mentioned before about the shift from single core to many core, so we're working very closely with silicon vendors to understand you know what do we need to do in our operating systems, our tools and services so that we can adopt this many core environment that silicon vendors are moving to. We also work with our silicon vendors on board support packages as well, especially around the CE operating system. We have system integrators that work with our OEMs in terms of getting to market with a number of these complex distributed systems. Software vendors that help in terms of developing these applications, and of course, hardware vendors that work on the underlying hardware.

Then for us, if you look at our business, the vast majority of our business is indirect, which means it goes through a distribution channel, so we have a very important distributor channel that we'll leverage as well. And if you want to find out more about these partners, I'd encourage you to go to our Web site, we've listed some of the logos on here so you can see some of the partners that we're working with, but I encourage you to go to our Web site and take a look at those partners, look at their expertise. I would reassure you that we put a lot behind these partners; if you look at our marketing

organization, we have a whole team focused on developing these partners and as we start to move forward into this service-oriented device environment, we're going to be working very closely with them to make sure that they have the knowledge so that they can support our customers in terms of getting to market with these devices.

So another piece of feedback that we had is that we need to be more transparent around our product road map, and what we're doing today is we're actually sharing a road map with you that expands out 24 months around our own embedded platform so that you as our customers and partners can really plan ahead. Starting off with CE, I've talked about the feature pack 5.0, we've talked about CE 6.0. We are ready to give you a sneak peek of the R2 release that's coming out in November, and then of course, we have CE V Next will be coming out, we'll give you more details on that at a later date. I think and hope we've given you a lot more clarity around our XPE platform and where we're going with that, the next major release coming out in 2008 and that is going to be based on the XP Service Pack 3, and we're going to be bringing in some of the advanced Vista technologies that our customers and partners have been asking for. And then a very important point on the V Next Plus One, this is where we're looking to offer the Vista componentized solution, again this is very, very different from the Vista with embedded restriction solution we have today. One is the full desktop solution for applications like ATMs, another one would be a componentized version where you can reduce the footprint, reduce the surface area for certain applications and systems.

And then in the Windows Embedded for point-of-service, we'll continue to innovate on that front as well. And we've highlighted it here as V Next and V Next Plus One, we will continue to innovate that; we'll communicate the functionality that we'll deliver. But what you'll see here is this common theme of major releases coming out every two years. In between, you'll see updates in terms of components to drive differentiation or updates in terms of tools as well. In addition to this product line, I explained as well that we've expanded it in terms of the .NET Micro Framework and then you have the full Vista implementation for certain applications. So this is the 24 month view of our road map, and I hope that this is beneficial to you and we'll continue moving forward and you have my commitment to make sure that we continue update you on where we're going with the products so you can plan your products moving forward.

In terms of next steps, we want to learn, browse, evaluate. So if you want to learn more about Windows CE 6.0 Release 2 we're going to give you a sneak peek of that in room 107. I'd definitely encourage you to go to that, ask lots of questions regarding Web Service on Devices. We're really excited about it

and we see that as the foundation for this new generation of service-oriented devices. Browse the Web, go to our Web site, and there you'll find a lot more information about the complete product line, the partner ecosystem that we have, and actually some of the success stories that we have as well with our customers. And then if you want to evaluate our products, just go to this site, where you go to evaluate these operating systems. We do have evaluation periods that you can use where you can go use the product, evaluate it against your next design, and then, of course, work with our partner ecosystem if you need to, to get support.

With that, we're going to continue our focus, and we'll be focusing on powering this next generation of smart, connected service-oriented devices. We're going to continue this ongoing investment. I've given you a peek into the kinds of investments that we've been making, I would argue that we're probably investing more in this area than any other software company in the industry. If you look across what we're doing on the Windows Embedded business, you look at the Microsoft Research team; we have the Live investments as well. We're going to continue to make huge investments in this area and we're making those investments so our partners and customers can rely on us and build out these types of solutions. We remain committed to leadership as well; we spend a lot of time thinking about where is the industry going. We realize it's about connected experiences and we realize what our role is going to be around platform tools and services that enable these smart, connected service-oriented devices. So we're going to continue to drive leadership in that area.

And then last but by no means least, we remain committed to the success of our partners and customers and we realize that this is a tremendous business opportunity for us, but it's an even bigger opportunity for our partners and customers and we need to make sure that we continue to deliver our products so that you can see success in this marketplace.

So with that, thank you very much and I appreciate your time.